

(11)Publication number : 2002-163310
(43)Date of publication of application : 07.06.2002

(51)Int.Cl.

G06F 17/50

(21)Application number : 2000-361456

(71)Applicant : HITACHI LTD

(22)Date of filing : 28.11.2000

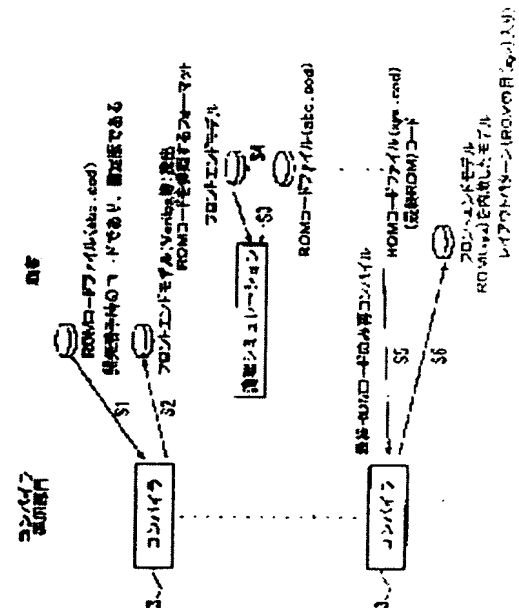
(72)Inventor : AIZAWA SHINYA
SHIMADA SHIGERU

(54) DESIGNING METHOD FOR SEMICONDUCTOR INTEGRATED CIRCUIT

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a designing method for semiconductor integrated circuit, which can easily transit to a corrected logic simulation and can easily manage the version of stored data as well when the stored data in a ROM module are corrected in logic design.

SOLUTION: When generating a data file expressing the ROM module up to a logic level while using a hardware description language, the information of bit data stored in the memory cell of the ROM module is stored in a file different from the data file of the module and in the form referring to the subordinate data file recorded with these bit data, the data file of the ROM module is generated. Further, the layout design of the mask patterns of the ROM module is executed so that the file name of the data file with the stored data recorded thereon can be visibly engraved on the inter-layer oxide film provided with a contact hole for determining the data value of the ROM module.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-163310

(P2002-163310A)

(43) 公開日 平成14年6月7日 (2002. 6. 7)

(51) Int.Cl. ⁷	識別記号	F I	テーマコード* (参考)
G 0 6 F 17/50	6 5 2	G 0 6 F 17/50	6 5 2 C 5 B 0 4 6
	6 5 4		6 5 4 K
			6 5 4 M
	6 5 6		6 5 6 R
	6 5 8		6 5 8 J

審査請求 未請求 請求項の数 5 O L (全 8 頁) 最終頁に続く

(21) 出願番号 特願2000-361456(P2000-361456)

(22) 出願日 平成12年11月28日 (2000. 11. 28)

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 藍澤 慎哉

東京都小平市上水本町五丁目20番1号 株式会社日立製作所半導体グループ内

(72) 発明者 島田 茂

東京都小平市上水本町五丁目20番1号 株式会社日立製作所半導体グループ内

(74) 代理人 100085811

弁理士 大日方 富雄

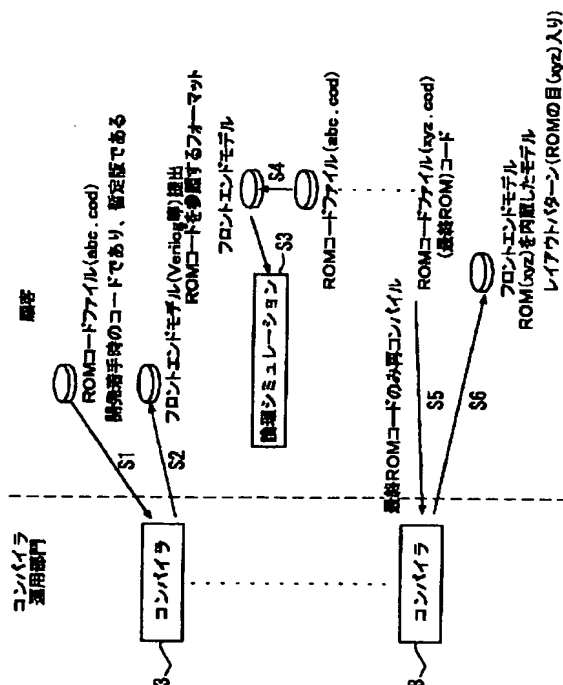
Fターム(参考) 5B046 AA08 BA04 JA05 KA02 KA03
KA06

(54) 【発明の名称】 半導体集積回路の設計方法

(57) 【要約】

【課題】 論理設計時にROMモジュールの格納データを修正した場合に、容易に修正後の論理シミュレーションへ移行することが可能であり、格納データのバージョン管理も容易な半導体集積回路の設計方法を提供することにある。

【解決手段】 ROMモジュールをハードウェア記述言語を用いて論理レベルまで表したデータファイルを生成する際に、ROMモジュールの記憶セルに記憶されるビットデータの情報をモジュールのデータファイルとは別のファイルに格納し、このビットデータが記録された副データファイルを参照する形式でROMモジュールのデータファイルを生成する。更に、ROMモジュールのデータ値を決定づけるコンタクトホールが設けられる層間酸化膜に、格納データが記録されたデータファイルのファイル名が視認可能な態様で刻み込まれるように、ROMモジュールのマスクパターンのレイアウト設計を行う。



【特許請求の範囲】

【請求項 1】 ROM モジュールをハードウェア記述言語を用いて論理レベルまで表したデータファイルを生成する際に、ROM モジュールの全部又は一部の記憶セルに記憶されるビットデータの情報をモジュールのデータファイルとは別のファイルに格納し、このビットデータが記録された副データファイルを参照する形式で ROM モジュールを論理レベルまで表したデータファイルを生成することを特徴とする半導体集積回路の設計方法。

【請求項 2】 半導体集積回路を製造するメーカー側と提供を受ける顧客側との共同により半導体集積回路の設計を行う半導体集積回路の設計方法であって、メーカーから顧客側に上記 ROM モジュールを論理レベルまで表したデータファイルを提供し、顧客側で上記 ROM モジュールを表したデータファイルに基づき論理シミュレーションを行うとともに、ROM モジュールに記憶するビットデータを修正して再度論理シミュレーションする場合に、顧客側で上記ビットデータが記録された副データファイルを変更して、上記 ROM モジュールを表すデータファイルに基づきビットデータ修正後の論理シミュレーションを行うことを特徴とする請求項 1 記載の半導体集積回路の設計方法。

【請求項 3】 顧客側で ROM モジュールに記憶するビットデータが確定した段階で該ビットデータをメーカー側に提示し、メーカーはこの確定したビットデータに基づき、各記憶セルに記憶されるビットデータの情報を含めた形式で ROM モジュールを論理レベルまで表したデータファイルを生成し、このデータファイルを顧客側に提供することを特徴とする請求項 2 記載の半導体集積回路の設計方法。

【請求項 4】 上記 ROM モジュールは設計時にワード長およびビット長を変更可能なモジュールであることを特徴とする請求項 1～3 の何れかに記載の半導体集積回路の設計方法。

【請求項 5】 ROM モジュールのマスクパターンのレイアウト設計をする際に、ROM モジュールの各記憶ビットのデータ値を決定づけるコンタクトホールが設けられる層間絶縁膜のモジュール配置部の近傍に、ROM モジュールに記憶されるビットデータが記録されたデータファイルを識別する識別情報が視認可能な態様で刻み込まれるように、マスクパターンのレイアウト設計を行うことを特徴とする半導体集積回路の設計方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 この発明は、ROM (Read Only Memory) モジュールを含んだ半導体集積回路の設計方法に適用して有用な技術に関し、例えば ASIC (Application Specific IC) などの半導体集積回路の設計方法に利用して特に有用な技術に関する。

【0002】

【従来の技術】 LSI 設計を行う分野では、LSI の製造後に手直しを行うと膨大なコストを要するため、設計段階でなるべくバグをなくしておきたいと云う要求が強い。そのため、設計段階でのバグを発見する手段として、設計された回路をコンピュータ上で模擬的に構成し動作させ、この模擬的な回路が所要の機能を果たすか否かを検証するシミュレーション手法が普及している。

【0003】 半導体集積回路のハードウェアの設計には、IC 全体のアーキテクチャやアルゴリズムの設計、機能設計、論理設計、電子回路設計、マスクパターンのレイアウト設計など種々のレベルがあるが、現在、これらの各レベルごとにシミュレーションを行ったり、また、これら複数のレベルを混合させた状態でシミュレーションを行って動作検証を行うことが可能になっている。

【0004】 メーカーと顧客側とで共同して LSI の設計・製造を行う ASIC の分野においては、顧客側とメーカー側の役割分担が明確に定められており、例えば、アーキテクチャやアルゴリズムの設計からゲートレベルの論理設計および論理シミュレーションまでを顧客側で行い、電子回路設計やマスクパターンのレイアウト設計以降をメーカー側で行うといった設計パターンがある。

【0005】 また、ASIC 設計では、ROM モジュールなどメーカー側から提供される機能回路（マクロセルとも云う）は、顧客にその機能設計データが示されない限り顧客側で論理合成することが出来ないため、一般に、論理シミュレーションへ移行する前に、次のような手順を必要としていた。

【0006】 すなわち、ASIC に ROM モジュールを組み込む場合、図 5 に示すように、顧客側からメーカー（コンパイラ運用部門）へ ROM モジュールの仕様（例えばビット長やワード長）や格納データが記録された ROM コードファイル（ファイル名「abc.cod」）を提示し、その後、これらの仕様やその格納データに基づきメーカー側のコンパイラ 3 で ROM モジュールの論理合成を行い、この論理合成後のモデル（以下、フロントモデルと称する）を顧客側に提供するという手順を必要としていた。そして、この手順の後に、顧客側で上記フロントモデルを用いて論理シミュレーションを行い、その動作結果から論理検証を行うことが可能となる。

【0007】

【発明が解決しようとする課題】 一般に、CPU (Central Processing Unit) や ROM モジュールを内蔵した CPU が ROM モジュールに格納されている命令コードを実行しながら動作するような LSI では、論理設計時に ROM モジュールの格納データの修正が頻繁に行われる。

【0008】 しかしながら、従来の ASIC の設計方法

では、図5に示すように、論理設計時にROMモジュールの格納データを修正して、修正後の論理シミュレーションを行う場合には、顧客側からメーカー側に修正後の格納データが記録されたROMコードファイル（「opq. cod」）を再度提示し、メーカー側で論理合成して修正後のフロントモデルを顧客側に提供し直してもらわなければ修正後の論理シミュレーションが行えなかった。この修正後のROMコードファイルの提示や再度の論理合成といった顧客とメーカー側とのやり取りは面倒で時間がかかるものであり、ROMモジュールの格納データの修正を繰り返し行えば、その分、上記ROMコードファイルの提示や論理合成を繰り返し行なわなければならない。

【0009】さらに、このようにROMモジュールの格納データを頻繁に修正した場合、最終版として確定された格納データがメーカー側に正しく示されていないと、メーカー側において最終版でない格納データでマスクパターンのレイアウト設計が行われてしまうといった恐れがある。また、マスクパターンに反映されているROMモジュールの格納データが最終版のものか確認するには、例えば各記憶セルのデータ値を決定するコンタクトホールが設けられる層間絶縁膜のパターンを視認し、このコンタクトホールのレイアウトと最終版の格納データとを逐一比較して手作業で確認するしかなかった。

【0010】この発明の目的は、論理設計時にROMモジュールの格納データを修正する場合でも、煩雑な手順を踏まずに容易に修正後の論理シミュレーションへ移行することが可能な半導体集積回路の設計方法を提供することにある。

【0011】この発明の他の目的は、ROMモジュールのマスクパターンに反映されている格納データのバージョン確認を容易に行うことのできる半導体集積回路の設計方法を提供することにある。

【0012】この発明の前記ならびにそのほかの目的と新規な特徴については、本明細書の記述および添付図面から明らかになるであろう。

【0013】

【課題を解決するための手段】本願において開示される発明のうち代表的なものの概要を説明すれば、下記のとおりである。

【0014】すなわち、ROMモジュールをハードウェア記述言語を用いて論理レベルまで表したデータファイルを生成する際に、ROMモジュールの全部又は一部の記憶セルに記憶されるビットデータの情報をモジュールのデータファイルとは別のファイルに格納し、このビットデータが記録された副データファイルを参照する形式でROMモジュールを論理レベルまで表したデータファイルを生成する半導体集積回路の設計方法である。

【0015】このような手段によれば、論理設計時にROMモジュールに記憶させるビットデータの修正を行っ

て再度論理シミュレーションを行なう場合でも、修正ごとにROMモジュールの論理合成を行なう必要がなくなり、ROMモジュールに格納されるビットデータを記録したデータファイルを修正するのみで、修正後の論理シミュレーションを行うことが出来る。従って、論理設計・論理シミュレーションのTAT（Turn Around Time）を短縮することが出来る。

【0016】望ましくは、半導体集積回路を製造するメーカー側と提供を受ける顧客側との共同により半導体集積回路の設計を行う半導体集積回路の設計方法であって、メーカーから顧客側に上記ROMモジュールを論理レベルまで表したデータファイルを提供し、顧客側で上記ROMモジュールを表したデータファイルに基づき論理シミュレーションを行うとともに、ROMモジュールに記憶するビットデータを修正して再度論理シミュレーションする場合に、顧客側で上記ビットデータが記録された副データファイルを変更して、上記ROMモジュールを表すデータファイルに基づきビットデータ修正後の論理シミュレーションを行うようにすると良い。

【0017】さらに望ましくは、顧客側でROMモジュールに記憶するビットデータが確定した段階で該ビットデータをメーカー側に提示し、メーカーはこの確定したビットデータに基づき、記憶セルに記憶されるビットデータの情報を含めた形式でROMモジュールを論理レベルまで表したデータファイルを生成して顧客側に提供すると良い。

【0018】このような手順によれば、最終的には、格納データの情報まで1つのデータファイルに包含されたROMモジュールの論理合成モデルが顧客側に示されるので、論理設計以降の過程で、ROMモジュールの格納データのバージョンを取り違えてしまうといった間違いを無くすることが出来る。

【0019】具体的には、上記ROMモジュールは設計時にワード長およびビット長を変更可能なモジュールである。

【0020】また、ROMモジュールのマスクパターンのレイアウト設計をする際に、ROMモジュールの各記憶ビットのデータ値を決定づけるコンタクトホールが設けられる層間絶縁膜のモジュール配置部の近傍に、ROMモジュールに記憶されるビットデータが記録されたデータファイルを識別する識別情報が視認可能な態様で刻み込まれるように、マスクパターンのレイアウト設計を行うと良い。

【0021】このような手段によれば、論理設計時にROMモジュールに格納するデータの修正を頻繁に行ない、マスクパターンのレイアウト設計後にマスクパターンに最終版が正しく反映されているかを確認したい場合でも、マスクパターンにあるビットデータが記録されたデータファイルの識別情報を視認することで、ROMモジュールに格納されるビットデータが最終版として確定

されたものか否か、容易に確認することが出来る。

【0022】また、この識別情報の配置を配線等のない領域に設定することで、半導体集積回路の製造後でも、上記識別情報を顕微鏡等を用いて視認し、ROMモジュールに記憶されているビットデータのデータファイルを確認することが出来る。

【0023】

【発明の実施の形態】以下、本発明の好適な実施例を図面に基いて説明する。

【0024】図1は、本発明を適用して好適な半導体集積回路の設計方法のうち論理設計時におけるROMモジュールに関する設計データ提供の流れを示す説明図である。同図においては、上から下に向かって処理が進められるようになっている。

【0025】この実施例の半導体集積回路の設計方法は、顧客とメーカーとが協同して設計を行なう例えばASIC (Application Specific IC) の設計過程で適用されるものであり、具体的には、ROMモジュールを内蔵するASICにおいて、ROMモジュールを含めたASIC全体の論理設計および論理シミュレーションを行なう過程に適用されるものである。この実施例では、ROMモジュールとして、設計時にビット長とワード長を所望の長さを選択できるコンパイルドタイプのマスクROMモジュールを使用している。

【0026】図1に示すように、この実施例の設計方法では、まず、ASICの開発初期の段階において、顧客側からASICに内蔵されるROMモジュールの基本仕様(ビット長とワード長など)と、ROMモジュールに格納する格納データ(ROMコード)が記録されたROMコードファイルの暫定版(ファイル名「a b d. c o d」)とがメーカー側に提示される(ステップS1)。

【0027】メーカー側に上記データが提示されると、メーカーのコンパイラ運用部門において、HDL等で記述され、データベースに予め登録されている当該仕様を有するROMモジュールの基礎データを読み出して、顧客側からのデータに基づきコンパイラ3を用いてROMモジュールの論理合成が行われる。以下、論理合成されたROMモジュールのことをフロントエンドモデルと呼ぶ。そして、このフロントエンドモデルが顧客側に提供される(ステップS2)。

【0028】この実施例では、上記コンパイラ3は、ネットリスト、RTL (register transfer level)、或いは論理式で入力された回路を、面積、遅延などの設計制約を満たすよう最適化したり、指定されたライブラリを使って論理合成を行なう論理合成ツールである。コンパイラ3からは、例えば、Verilog (登録商標) HDL (hardware design language) などのハードウェア記述言語を用いて回路を論理レベルまで記述したデータファイルが生成され、顧客側では、このデータファイルを用いて回路の論理シミュレーションを行なうことが

可能となる。

【0029】この実施例においては、初期段階で顧客側に提供されるフロントエンドモデルは、ROMモジュールに格納される格納データ(ROMコード)が記録された副データファイルを外付けにして、上記格納データの情報が除かれたROMモジュールのフロントエンドモデルからこの副データファイルを参照する形式とされる。以下、このモデルのことを参照型フロントエンドモデルと呼ぶ。

10 【0030】なお、この実施例では、コンパイラ3による論理合成により、ROMコードファイルの内容が格納される副データファイルと、ROMコードファイルの内容が除かれたフロントエンドモデルとの両方が生成される形態になっているが、コンパイラ3からは上記参照型のフロントエンドモデルだけ生成し、副データファイルは所定のフォーマットに従って顧客側で作成するようにしても良い。また、このような形態の場合には、顧客側からメーカー側への暫定的なROMコードファイルの提示を省くことも出来る。

20 【0031】図2には、上記参照型フロントエンドモデルの具体的な記述の一例を示す。

【0032】同図中、範囲aの内容は内部変数の時間軸の刻みやバージョン名など、モジュール記載の始まりを参照的に示すものである。b1行の内容はモジュール名と該モジュールにおいて使用される入出力変数、b2行の内容はROMモジュールのビット長やワード長などのパラメータ、b3行の内容はROMモジュールの入力信号をそれぞれ示している。また、範囲cの内容はモジュール内部で使用する信号名と外部入力される信号名との対応を、範囲dの内容はROMモジュールの動作記述を示している。

30 【0033】また、範囲eには、指定されたデータファイルのデータを参照するコマンドコードや、参照先のデータファイル名e1、参照したデータが代入される配列変数名“MEM”やその始まりの配列番号“0”などが示されている。ここで参照されるデータは、ROMモジュールに格納される格納データ(ROMコード)であり、該格納データを欄e1に示されるデータファイルに記録しておくことで、該格納データがモジュール内の配列MEM(9'b000000000)~MEM(9'b111111111)に代入されるようになっている。なお、「9'b」とは9ビットの値が次に続くことを示している。

40 【0034】また、範囲fは、ROMモジュールの記述の終りを示している。

【0035】参照型フロントエンドモデルが提供されたら、顧客側でASICの論理シミュレーションを行ない(ステップS3)、その動作結果に基づき論理検証を行って所望の機能が得られない箇所について論理設計の修正や、ROMモジュールに格納するファームウェアの修

正などを行う。

【0036】そして、ROMモジュールの格納データを修正した場合には、顧客側でこの修正を図2の欄e 1に示されるデータファイルに反映させる（ステップS 4）。それにより、コンパイラ3によるROMモジュールの再度の論理合成を行なうことなく、この修正を論理合成後のフロントエンドモデルに反映させることが出来る。そして、修正が反映されたフロントエンドモデルを用いて、修正後の論理シミュレーションを行なう（ステップS 3）。

【0037】上記のような論理シミュレーション、論理検証、並びに、ファームウェアの修正を繰り返して、最終的な格納データが確定したら、この格納データが記録された最終版のROMコードファイル（ファイル名「xyz.cod」）を顧客側からメーカー側に提示する（ステップS 5）。

【0038】最終版のROMコードファイルが提示されたら、メーカーのコンパイラ運用部門において、初期段階で提示されたROMモジュールの仕様および最終版のROMコードファイルに基づき、コンパイラ3を用いてROMモジュールの論理合成を行ない、このフロントエンドモデルを顧客側に提供する（ステップS 6）。ここで提供されるフロントエンドモデルは、ROMモジュールの格納データ（ROMコード）も1つのデータファイルに包含した形式のものである。以下、このモデルのことを内蔵型フロントエンドモデルと呼ぶ。

【0039】図3には、上記内蔵型フロントエンドモデルの具体的な記述の一例を示す。

【0040】この内蔵型フロントエンドモデルの形式は、従来、一般的に用いられているものである。図3において、範囲a～d、fの内容は図2の参照型フロントエンドモデルのものと同一のものである。

【0041】範囲gでは、ROMモジュールに格納される格納データが配列変数“MEM(9'b0000000000)”～“MEM(9'b1111111111)”に代入されるようになっている。すなわち、この格納データが最終版のROMコードファイルのデータであり、最終版のROMコードファイルがフロントエンドモデルに内蔵された形式となっている。

【0042】また、メーカー側のコンパイラ運用部門では、内蔵型のフロントエンドモデルの作成とともに、ROMモジュールのマスクパターンのレイアウト設計も行なう。

【0043】図4には、半導体チップのレイアウトパターンの概要図を示す。同図(a)は半導体チップ全体の中のROMモジュールの配置を、(b)はROMモジュールの部分を示している。

【0044】ROMモジュール10は、ROMの記憶セルがマトリックス状に配置された記憶セルアレイ11や、Yデコーダやセンスアンプなどの周辺回路等を備え

たものである。また、特に制限されるものでないが、この実施の形態のROMモジュール10は、ビット線とワード線の各交点の必要な位置にMOSトランジスタを配置して記憶セルを構成し、配線工程においてビット線とMOSトランジスタのドレイン若しくはソースを接続又は非接続とすることで、各記憶セルに記憶されるデータ値が“1”又は“0”に設定されるタイプのマスクROMである。つまり、MOSトランジスタとビット線とを接続する層間絶縁膜のコンタクトホールh(viaホールとも呼ぶ)の有無により各記憶セルのデータ値が決定されるものである。

【0045】さらに、この実施の形態では、図4(b)に示すように、ROMモジュールのレイアウト設計を行なうときに、コンタクトホールhが形成される層間絶縁膜において他の配線接続に影響を及ぼさない箇所、例えば、ROMモジュールの周縁を縁取るように配される電源配線12と重なる範囲に、このレイアウト設計に組み込まれたROMコードファイルを識別するための情報としてそのファイル名「xyz」がコンタクトホールhによりドット表示されるように、コンタクトホールhのレイアウト設計を行う。

【0046】そして、このようなレイアウト設計を行ったら、コンタクトホールhのレイアウトパターンが示されたパターンモデルを、上記フロントエンドモデルとともに顧客側に提供する(図1のステップS 6)。そして、これら内蔵型フロントエンドモデルやパターンモデルに基づき顧客側で最終的な確認作業が行われる。

【0047】以上のように、この実施例の半導体集積回路の設計方法によれば、論理設計時にROMモジュールの格納データの修正やその論理シミュレーション並びに論理検証を繰り返し行なう場合でも、格納データの修正ごとにROMモジュールの論理合成をメーカー側に依頼して行なうという面倒で煩雑な工程が省けるので、メーカー側および顧客側の工数が削減され、論理設計におけるTATの短縮、延いてはASIC開発期間の短縮を図ることが出来る。

【0048】また、最終的には、格納データの情報まで1つのデータファイルに包含されたROMモジュールの論理合成モデルが顧客側に提供されるので、論理設計以降の過程で、ROMモジュールの格納データのバージョンを取り違えてしまうといった間違いを無くすることが出来る。

【0049】また、ROMモジュールのマスクパターンのレイアウト設計をする際に、ROMモジュールの各記憶ビットのデータ値を決定づけるコンタクトホールhが設けられる層間絶縁膜に、ROMモジュールの格納データが記録されたデータファイルのファイル名が視認可能な態様で刻み込まれるように、マスクパターンのレイアウト設計が行われるので、マスクパターンのレイアウト設計後にマスクパターンにROMコードファイルの最終

版が正しく反映されているかを確認したい場合に、パターンモデルに表示されたROMコードファイルのファイル名を視認するだけで容易に確認することが出来る。

【0050】また、このファイル名を表すコンタクトホールhの配置を配線等のない領域に設定することで、半導体集積回路の製造後でも、上記ファイル名を顕微鏡等を用いて視認し、ROMモジュールの格納されているROMコードファイルのバージョン等を確認することが出来る。

【0051】以上本発明者によってなされた発明を実施例に基づき具体的に説明したが、本発明は上記実施例に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。

【0052】例えば、半導体集積回路に組み込むROMモジュールとして、コンパイルドタイプのROMモジュールを例示したが、ビット長やワード長が固定のROMモジュールであっても同様の効果が得られる。また、最終的にROMモジュールの格納データも一体化された内蔵型のフロントエンドモデルを顧客側に提供するようにしたが、この内蔵型フロントエンドモデルの論理合成や顧客への提供を省くことも出来る。

【0053】以上の説明では主として本発明者によってなされた発明をその背景となった利用分野であるASICの設計方法について説明したがこの発明はそれに限定されるものでなく、ROMモジュールを内蔵する種々の半導体集積回路の設計方法に広く利用することができる。

【0054】

【発明の効果】本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記のとおりである。

【0055】すなわち、本発明に従うと、論理設計時にROMモジュールの格納データの修正やその論理シミュレーション並びに論理検証を繰り返し行なう場合でも、格納データの修正ごとにROMモジュールの論理合成をメーカー側に依頼して行なうという面倒で煩雑な工程が省けるので、メーカー側および顧客側の工数が削減され、論理設計におけるTATの短縮、延いてはASIC

開発期間の短縮を図ることが出来るという効果がある。

【0056】また、ROMモジュールのマスクパターンレイアウト設計をする際に、ROMモジュールの各記憶ビットのデータ値を決定づけるコンタクトホールが設けられる層間絶縁膜に、ROMモジュールの格納データが記録されたデータファイルのファイル名が視認可能な態様で刻み込まれるように、マスクパターンのレイアウト設計が行われるので、マスクパターンのレイアウト設計後にマスクパターンにROMコードファイルの最終版が正しく反映されているかを確認したい場合に、パターンモデルに表示されたROMコードファイルのファイル名を視認するだけで容易に確認することが出来るという効果がある。

【図面の簡単な説明】

【図1】本発明を適用して好適な半導体集積回路の設計方法のうち論理設計時におけるROMモジュールに関する設計データ提供の流れを示す説明図である。

【図2】設計初期の段階でメーカーから顧客側に提供されるデータ参照型のフロントエンドモデルの具体的な記述例を示す参考図である。

【図3】ROMモジュールの格納データが確定した段階でメーカーから顧客側に提供されるデータ内蔵型のフロントエンドモデルの具体的な記述例を示す参考図である。

【図4】半導体チップのレイアウトパターンの概要を示すもので、(a)は半導体チップ全体の中のROMモジュールの配置、(b)はROMモジュールの部分拡大した図である。

【図5】従来の半導体集積回路の設計方法で、論理設計時におけるROMモジュールに関するデータ提供の流れを示す説明図である。

【符号の説明】

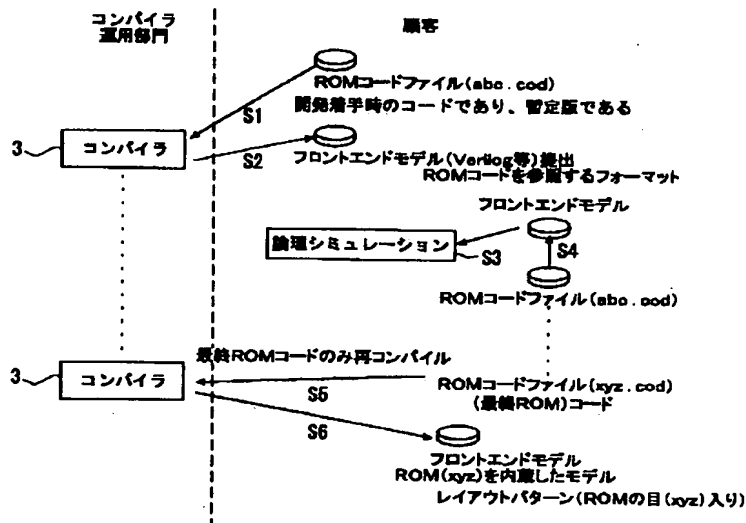
10 ROMモジュール

12 電源配線

h コンタクトホール

e ROMコードファイルを外部から参照させるコードの記述範囲

【図 1】



【図 2】

```

timescale 1ns / 10ps 'delay_mode_path
// 78C CROM verilog model version 0.8 2000,04,11
'suppress_faults 'enable_portfaults
'celldefine
} a

module rm051216 (a,q,ck,bs,resb);
parameter numAddr=9, bits =16, word_depth=512;
input ck,bs,resb;
buf(CK,ck);
buf(BS,bs);
buf(RESB,resb);
} b1
} b2
} b3
} c
} d

initial begin
$readmemb("rm051216.cod",MEM,0);
end
} e ← ファイル名の指定

endmodule
} f

'endcelldefine
'disable_portfaults
'nosuppress_faults
} g

```

【図 3】

```

timescale 1ns / 10ps 'delay_mode_path
// 78C CROM verilog model version 0.8 2000,04,11
'suppress_faults 'enable_portfaults
'celldefine
} a

module rm051216 (a,q,ck,bs,resb);
parameter numAddr=9, bits =16, word_depth=512;
input ck,bs,resb;
buf(CK,ck);
buf(BS,bs);
buf(RESB,resb);
} b1
} b2
} b3
} c
} d

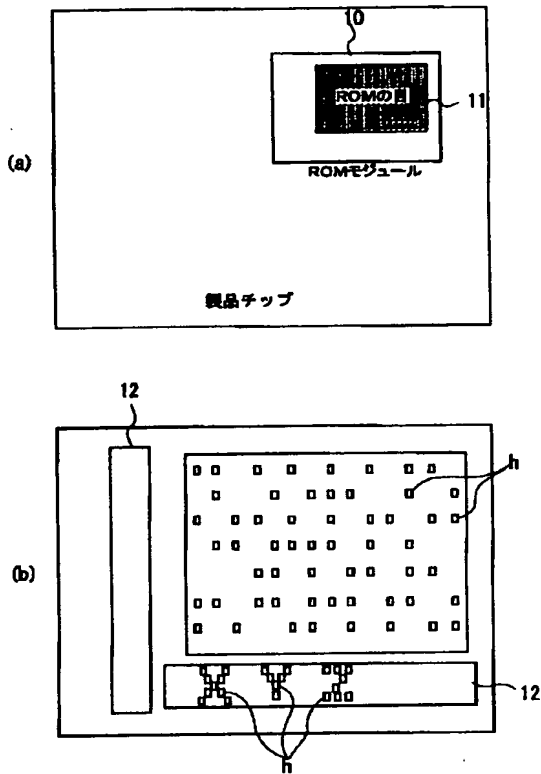
initial begin
MEM[9'b000000000]=16'b1111111111111111;
MEM[9'b000000001]=16'b1111111111111111;
MEM[9'b000000010]=16'b1111111111111111;
MEM[9'b111111111]=16'b0110011001100110;
end
} e

endmodule
} f

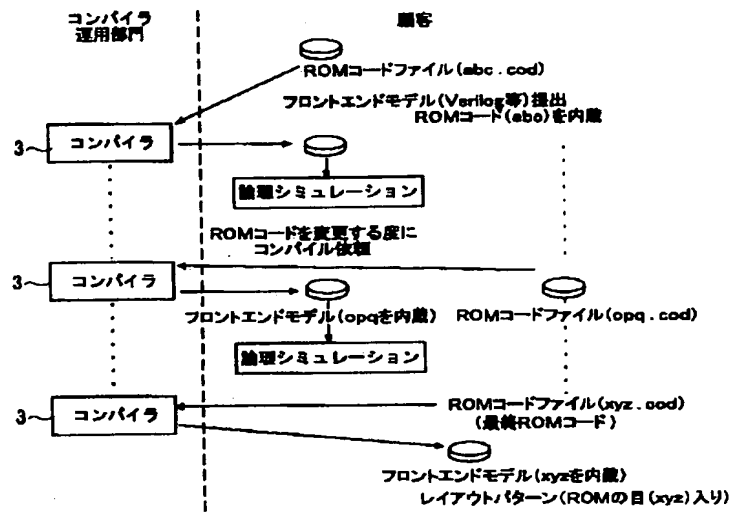
'endcelldefine
'disable_portfaults
'nosuppress_faults
} g

```


【図4】



【図5】



フロントページの続き

(51) Int. Cl.⁷

G 0 6 F 17/50

識別記号

6 5 8

F I

G 0 6 F 17/50

特コード (参考)

6 5 8 M

BEST AVAILABLE COPY